intel

product release notes

D86ASM51NLR2.4

These release notes are divided into the following sections:

MANUAL UPDATE

TOOL-SPECIFIC NOTES

DOS OPERATION-SPECIFIC NOTES

MANUAL UPDATE

The following information should be added to the *MCS®-51 Utilities User's Guide for DOS Systems* manual:

OH FOR MICROCONTROLLERS

The OH is a tool included with the RL-51 for 8051 products. OH is an object file converter that converts an absolute 8051 object file to a hex file. These hex files can be used with tools that do not accept Intel standard object file formats.

Invocation Syntax

The invocation syntax for OH is:

```
OH objfilename [TO hexfilename]
```

Where:

objfilename	is an absolute 8051 object file
hexfilename	is the file containing the hexadecimal output

If a *hexfilename* is not specified, the output file name defaults to the root of the object file name with a .hex extension.

Examples:

The following example converts the absolute 8051 object file and placed the output into a hexadecimal format file.

```
OH savel.obj TO save.h51
```

The next example does not specify a hexadecimal output filename.

```
OH tot2.obj
```

The hexadecimal file is now called tot2.hex

Output File

The following is a sample hex output:

```
:10000300850A83850B827401F02275080275090045
:0800130085080A85090B1103A1
:0300000002001BE0
:09001B0075810C75D00002000D86
:00000001FF
```

In the output file, the record format is:

:	length	offset	type	content	checksum

Where:

: is the record header

length are the next two hex digits representing the record length. OH outputs records of 16 bytes or less, causing most records to start with :10

offset are the next four hex digits representing the offset from the record's beginning

type are the next two hex digits 00 representing the record type

content are the next two hex digits representing each content byte

checksum are the remaining two hex digits

Each record in the file contains 16 bytes. If the information available for a record is greater than 16 bytes, OH divides it into 16 byte sections. If there are remaining bytes after this operation, they are combined with bytes left over from adjacent records in memory. These adjacent bytes in the OH buffer, may be from previous, or following records. If there are no adjacent bytes available for combination, OH puts the extra bytes on a new line.

The hexidecimal file ends with the following module end record:

```
:00000001FF
```

Error Messages

OH error messages are fatal errors. Processing of the object file is stopped and one of the following error messages is issued:

```
*** ERROR - invocation should be: OH <infile> [TO outfile>]
```

The invocation syntax is incorrect. OH expected the TO keyword but found something else. Re-invoke OH using the correct syntax

```
*** ERROR - input is not an absolute object file
```

The object file is not absolutely located.

```
*** ERROR - input has a record longer than 32K, sorry
```

Make sure the records in the file are less than 32K bytes

```
*** ERROR - on reading OBJECT: I/O error in invalid object
      file
```

OH has detected an invalid format. Be sure the object file is available, and that it is an absolutely located file.

Errors

In appendix B of the *MCS®-51 Utilities User's Guide for DOS Systems*, order number 122747 several error messages are missing. Error messages 114-120, 122-127 and fatal errors 201-207 are not listed. Following are the missing error messages:

```
ERROR 114: SEGMENT DOES NOT FIT
SEGMENT:          segment-name, base, length
```

The specified segment cannot be located at the base specified by the locating control. Starting at that base address there is insufficient memory for a segment of its length. The specified segment is ignored.

```
ERROR 115: INPAGE SEGMENT IS GREATER THAN 256 BYTES
SEGMENT:          segment-name
```

The specified INPAGE segment is greater than one page. The specified segment is ignored.

```
ERROR 116: INBLOCK SEGMENT IS GREATER THAN 2047 BYTES
SEGMENT:          segment-name
```

The specified INBLOCK segment is greater than one block. The specified segment is ignored.

```
ERROR 117: BIT ADDRESSABLE SEGMENT IS GREATER THAN 16 BYTES
SEGMENT:          segment-name
```

The specified BIT-ADDRESSABLE segment is greater than the bit space. The specified segment is ignored.

```
ERROR 118: REFERENCE MADE TO ERRONEOUS EXTERNAL
SYMBOL:           external-name
MODULE:           file-name (module-name)
REFERENCE:        code-address
```

The specified, ignored external symbol is referenced in the specified module at the specified code address.

```
ERROR 119: REFERENCE MADE TO ERRONEOUS SEGMENT
SYMBOL:          segment-name
MODULE:          file-name (module-name)
REFERENCE:       code-address
```

A symbol, which is defined using the specified, but ignored, segment, is referenced in the specified module at the specified code address.

```
ERROR 120: CONTENT BELONGS TO ERRONEOUS SEGMENT
SYMBOL:          segment-name
MODULE:          file-name (module-name)
```

A content record, which belong to the specified, but ignored, segment, has been encountered. The content record is not relocated.

```
ERROR 122: CANNOT FIND MODULE
MODULE:          file-name (module-name)
```

The specified module name, which was explicitly requested from the specified file (in the command tail), was not found in that file.

```
ERROR 123: ABSOLUTE IDATA SEGMENT DOES NOT FIT
MODULE:          file-name (module-name)
FROM:            data-address
TO:              data-address
```

The specified module contains an absolute IDATA segment that occupies non-existent internal RAM space in the target machine. The segment is ignored. Notice, however, that the module may contain erroneous references to this segment, which are not reported.

```
ERROR 124: RESERVED
```

```
ERROR 125: MORE ERRORS ENCOUNTERED NOT REPORTED
```

Non-fatal errors encountered henceforth will not be reported.

```
ERROR 126: OVERLAY MODULE NOT FOUND
MODULE:          file-name (module-name)
```

The specified module name explicitly mentioned in the overlay control was not found.

ERROR 126: OVERLAY DATA ADDRESS SPACE OVERFLOW
SPACE: *on-chipRAM space*

RL-51 was unable to allocate an overlaid segment of the specified address space. Try to link with the NOOVERLAY control.

Fatal Errors

FATAL ERROR 201: INVALID COMMAND LINE SYNTAX
partial command

A syntax error was detected in the command. The command is repeated up to and including the point of error.

FATAL ERROR 202: INVALID COMMAND LINE; TOKEN TOO LONG
partial command

A command line contains a token that is too long. The command is repeated up to and including the point of error.

FATAL ERROR 203: EXPECTED ITEM MISSING
partial command

An expected item in the command line, such as an input file name or a file name following the TO is missing. The command is repeated up to and including the point of error.

FATAL ERROR 204: INVALID KEY WORD
partial command

An invalid key word was found in the command. The command is repeated up to and including the point of error.

FATAL ERROR 205: NUMERIC CONSTANT TOO LARGE
partial command

A numeric constant greater than 0FFFH was found in the command. The command is repeated up to and including the point of error.

FATAL ERROR 206: INVALID CONSTANT
partial command

An illegally constructed context was found. A common example of this error is entering a hexadecimal number with a letter first. The command is repeated up to and including the point of error.

FATAL ERROR 207: INVALID NAME
partial command

An illegally constructed name was found. Names can be from 1 through 40 characters long and must be composed of the letters A-Z, the digits 0-9, or special characters (?,@,_). The first character must be a letter or a special character. The command is repeated up to and including the point of error.

There are some errors in the *MCS®-51 Macro Assembler User's Guide for DOS Systems*, order number 122752. **These errors are:**

Disregard any reference to a manual called the *MCS-51 Family Of Single-Chip Microcomputer User's Guide*, order number 121517. This manual is no longer available.

Disregard any reference to a manual called the *Universal PROM Programmer Reference Manual*, order number 9800133. This manual is no longer available.

In Chapter 7, "Assembler Output: Error Messages and Listing File Format", the message for ERROR 34 is incorrect. It should read, "The maximum number of external symbols referred to by a single module (255) has been exceeded."

In Appendix J a cross-reference to Chapter 6 is made regarding a complete description of all the error messages. The cross-reference should be to Chapter 7.

TOOL-SPECIFIC NOTES

CROSS REFERENCE HEADING

The cross reference heading in the list file does not conform to the PAGEWIDTH control.

CROSS REFERENCE LISTING

Single character symbols have an additional character appended in the cross reference listing. The problem occurs only when the NAME is greater than 30 characters.

Example:

```
NAME  The_appended_character_name_program
EXTERN DATA (A,B,C,ASKIT)
CSEG AT 0
END
```

To correct the cross reference listing, remove characters from the NAME directive.

PDF FILE CHANGES

The assembler is distributed with Processor Definition Files (PDF) for members of the MCS-51 family. Each of the PDF files contains definitions of the special function registers of one MCS-51 member. These files are:

REG44.PDF
REG51.PDF
REG52.PDF
REG152.PDF
RG51FA.PDF
RG51FB.PDF
REG451.PDF
REG452.PDF

For more information about PDF files see Appendix L of the *MCS®-51 Macro Assembler User's Guide for DOS Systems.*

SYMBOL TABLE LIMITATIONS

ASM51	1314 six character symbols or with the NOMACRO control 2400 six character symbols.
RL51	2600 ten character symbols.

DOS OPERATION-SPECIFIC NOTES

DIRECTING OUTPUT TO A FULL DISK

When directing output to a full disk, an Intel translator or Relocation and Linkage (R & L) tool may terminate prematurely without giving any error message. Make sure your disk has sufficient space to contain any output these tools may generate before invoking them.

FILE SHARING CONFLICTS

File sharing conflicts may occur when using an Intel translator or R & L tool in a network environment. Before invoking an Intel translator or R & L tool (with network support), invoke the DOS V3.0 or later SHARE command. It is recommended that you invoke the SHARE command in your AUTOEXEC.BAT file.

USER-DEFINED LOGICAL NAMES

Confusion may occur on user-defined logical names. The default assignments for :F0: through :F9: and :WORK: are not in the user documentation. If these logical devices are not defined with the SET command, the default assignments for :F0: through :F9: are to devices A through J. The default assignment for :WORK: is the current default disk. Use the SET command to assign the desired logical devices.

USING CTRL-BREAK VERSUS CTRL-C

CTRL-C does not work as the interrupt character like CTRL-BREAK. Use CTRL-BREAK instead of CTRL-C.

FILES CONSISTING OF AN 8-DIGIT HEX NUMBER

Files consisting of an 8-digit hex number with no extension may be left in the current :WORK: directory after typing CTRL-BREAK to abort an Intel translator or R & L tool or a user program converted to DOS with UDI2DOS.EXE. These are temporary files created by these programs to store intermediate data. They are normally deleted at the end of a program's normal execution. Delete or ignore the files. No important information is contained in them.

FATAL ERROR SPECIFYING LONG PATHNAMES

A fatal error is flagged when specifying long pathnames for output files, and the translator or R & L tool aborts. While the DOS manual indicates that the maximum number of characters in a pathname is 63, in practice various products seem to restrict pathnames to less than 63 characters. To ensure compatibility with all products, make sure that all output pathnames do not exceed 43 characters.

ERROR #21 --- FILE DOES NOT EXIST

This message is issued by the Operating System even though the file does exist. The PC/DOS Operating System is installed incorrectly. Re-install the Operating System and make sure that it is DOS V3.0 or later. DOS V3.0 or greater has a different COMMAND.COM file.

SHARING VIOLATION ERROR READING DRIVE

This message is returned when a program, which has been invoked by a command file, is interrupted in the middle of that process by a system reset and you later attempt to access that file. If you have directed output from that command file, the same error also appears for that output file. This does not occur on Intel NRM file servers but may occur on other types of networks. A system reset may leave these files open, thus they cannot be reopened. Avoid resetting the system while a command file is executing on the network. You can type CTRL-BREAK's until the command file has exited.

INPUT/OUTPUT FILE CONTENTS MAY BE LOST

The contents of a file may be lost when the same file is specified for input and output to a translator or R & L tool. Depending on the order in which a translator or R & L tool opens the input and output files, the file may be overwritten if it is used for both input and output. When a file is opened for output on DOS, any previous contents are lost unless the file is already opened for input and the SHARE command is in effect. When invoking a translator or R & L tool under DOS, ensure that the output filename differs from the input filename.

DOS ERRORLEVEL

The MCS-51 assembler sets the DOS ERRORLEVEL variable as follows:

0 No warnings or errors
1 Warnings occurred
2 Errors occurred
3 Fatal Errors occurred

TABLE OF CONTENTS

	PAGE
Program Status Word	1
Operand Definitions	1
Arithmetic Instructions	2
Logical Instructions	3
Data Move Instructions	4
Bit Manipulation Instructions	5
No Operation	5
Program and Machine Control Instructions	6
Codes for Instructions Using Registers	7
Predefined Byte Addresses	8
Predefined Code Addresses	8
Predefined Bit Addresses	9
Macro Processor Language (MPL) Functions	10
Assembly Time Expression Operators	11
Assembler Directives	11
Table of Instruction Opcodes in Hexadecimal Order	12
Hex-ASCII Table	18

PROGRAM STATUS WORD (PSW—0D0H)

C	AC	F0	RS1	RS0	OV		P
7	6	5	4	3	2	1	0

C	Carry Flag
AC	Auxiliary Carry Flag
F0	Flag 0
RS1	Register Bank Select
RS0	Register Bank Select
OV	Overflow Flag
PSW.1	Not Used
P	Parity

OPERAND DEFINITIONS

Operand	Meaning
#data	data coded in instruction
data addr	on-chip memory address
Rr	0 ≤ r ≤ 7 General-Purpose register
@Rr	0 ≤ r ≤ 1 Indirect address register
code addr	16-bit address encoded as: full 16-bit 11-bit page address 8-bit relative offset
A	Accumulator
C	Carry flag
bit addr	bit address (on-chip)
/bit addr	complemented contents of bit address
DPTR	Data Pointer
PC	Program Counter
AB	Register Pair

ARITHMETIC INSTRUCTIONS

Hex Code	Mnemonic	Operands
24	ADD	A,*#data*
26	ADD	A,@R0
27		R1
2*	ADD	A,R*r*
25	ADD	A,*data addr*
34	ADDC	A,*#data*
36	ADDC	A,@R0
37		R1
3*	ADDC	A,R*r*
35	ADDC	A,*data addr*
94	SUBB	A,*#data*
96	SUBB	A,@R0
97		R1
9*	SUBB	A,R*r*
95	SUBB	A,*data addr*
04	INC	A
06	INC	@R0
07		R1
0*	INC	R*r*
05	INC	*data addr*
A3	INC	DPTR
14	DEC	A
16	DEC	@R0
17		R1
1*	DEC	R*r*
15	DEC	*data addr*
A4	MUL	AB
84	DIV	AB
D4	DA	A

*See page 7 for instruction's hex code.

LOGICAL INSTRUCTIONS

Hex Code	Mnemonic	Operands
54	ANL	A,*#data*
56	ANL	A,@R0
57		R1
5*	ANL	A,R*r*
55	ANL	A,*data addr*
53	ANL	*data addr*,*#data*
52	ANL	*data addr*,A
44	ORL	A,*#data*
46	ORL	A,@R0
47		R1
4*	ORL	A,R*r*
45	ORL	A,*data addr*
43	ORL	*data addr*,*#data*
42	ORL	*data addr*,A
64	XRL	A,*#data*
66	XRL	A,@R0
67		R1
6*	XRL	A,R*r*
65	XRL	A,*data addr*
63	XRL	*data addr*,*#data*
62	XRL	*data addr*,A
E4	CLR	A
F4	CPL	A
23	RL	A
33	RLC	A
03	RR	A
13	RRC	A
C4	SWAP	A

* See page 7 for instruction's hex code.

DATA MOVE INSTRUCTIONS

Hex Code	Mnemonic	Operands
74	MOV	A,#*data*
E6	MOV	A,@R0
E7		R1
E*	MOV	A,R*r*
E5	MOV	A,*data addr*
F6	MOV	@R0,A
F7		R1
76	MOV	@R0,#*data*
77		R1
A6	MOV	@R0,*data addr*
A7		R1
F*	MOV	R*r*,A
7*	MOV	R*r*,#*data*
A*	MOV	R*r*,*data addr*
F5	MOV	*data addr*,A
75	MOV	*data addr*,#*data*
86	MOV	*data addr*,@R0
87		R1
8*	MOV	*data addr*,R*r*
85	MOV	*data addr*,*data addr*
90	MOV	DPTR,#*data* *
93	MOVC	A,@A + DPTR
83	MOVC	A,@A + PC
E0	MOVX	A,@DPTR
E2	MOVX	A,@R0
E3		R1
F0	MOVX	@DPTR,A
F2	MOVX	@R0,A
F3		R1
C0	PUSH	*data addr*
D0	POP	*data addr*
C **	XCH	A,R*r*
C5	XCH	A,*data addr*
C6	XCH	A,R0
C7		R1
D6	XCHD	A,@R0
D7		R1

* 16-bit data operand.

** See page 7 for instruction's hex code.

BIT MANIPULATION INSTRUCTIONS

Hex Code	Mnemonic	Operands
C3	CLR	C
C2	CLR	*bit addr*
D3	SETB	C
D2	SETB	*bit addr*
B3	CPL	C
B2	CPL	*bit addr*
82	ANL	C,*bit addr*
B0	ANL	C,/*bit addr*
72	ORL	C,*bit addr*
A0	ORL	C,/*bit addr*
A2	MOV	C,*bit addr*
92	MOV	*bit addr*,C

NO OPERATION

Hex Code	Mnemonic
00	NOP

PROGRAM AND MACHINE CONTROL INSTRUCTIONS

Hex Code	Mnemonic	Operands
†1	AJMP	*code addr*
B6	CJNE	@R0,#*data*,*code addr*
B7		R1
B4	CJNE	A,#*data*,*code addr*
B5	CJNE	A,*data addr*,*code addr*
B*	CJNE	Rr,#*data*,*code addr*
D*	DJNZ	Rr,*code addr*
D5	DJNZ	*data addr*,*code addr*
20	JB	*bit addr*,*code addr*
10	JBC	*bit addr*,*code addr*
40	JC	*code addr*
73	JMP	@A+DPTR
30	JNB	*bit addr*,*code addr*
50	JNC	*code addr*
70	JNZ	*code addr*
60	JZ	*code addr*
02	LJMP	*code addr*
80	SJMP	*code addr*
†1	ACALL	*code addr*
12	LCALL	*code addr*
22	RET	
32	RETI	
**	JMP	*code addr*
**	CALL	*code addr*

† First three bits of the opcode are formed by the code address operand

* See page 7 for instruction's hex code.

** Generic form of instruction.

CODES FOR INSTRUCTIONS USING REGISTERS

	MOV Rr,A	MOV Rr, #*data*	MOV A,Rr	MOV *data addr*,Rr
R0	F8H	78H	E8H	88H
R1	F9H	79H	E9H	89H
R2	FAH	7AH	EAH	8AH
R3	FBH	7BH	EBH	8BH
R4	FCH	7CH	ECH	8CH
R5	FDH	7DH	EDH	8DH
R6	FEH	7EH	EEH	8EH
R7	FFH	7FH	EFH	8FH
	XCH A,Rr	XRL A,Rr	INC Rr	MOV Rr, *data addr*
R0	C8H	68H	08H	A8H
R1	C9H	69H	09H	A9H
R2	CAH	6AH	0AH	AAH
R3	CBH	6BH	0BH	ABH
R4	CCH	6CH	0CH	ACH
R5	CDH	6DH	0DH	ADH
R6	CEH	6EH	0EH	AEH
R7	CFH	6FH	0FH	AFH
	ADD A,Rr	ADDC A,Rr	ANL A,Rr	CJNE Rr, #*data*, *code addr*
R0	28H	38H	58H	B8H
R1	29H	39H	59H	B9H
R2	2AH	3AH	5AH	BAH
R3	2BH	3BH	5BH	BBH
R4	2CH	3CH	5CH	BCH
R5	2DH	3DH	5DH	BDH
R6	2EH	3EH	5EH	BEH
R7	2FH	3FH	5FH	BFH
	ORL A,Rr	SUBB A,Rr	DEC Rr	DJNZ Rr, *code addr*
R0	48H	98H	18H	D8H
R1	49H	99H	19H	D9H
R2	4AH	9AH	1AH	DAH
R3	4BH	9BH	1BH	DBH
R4	4CH	9CH	1CH	DCH
R5	4DH	9DH	1DH	DDH
R6	4EH	9EH	1EH	DEH
R7	4FH	9FH	1FH	DFH

PREDEFINED BYTE ADDRESSES

Symbol	Hex Address	Meaning
ACC	E0	ACCUMULATOR
B	F0	MULTIPLICATION REGISTER
DPH	83	DATA POINTER *<high byte>*
DPL	82	DATA POINTER *<low byte>*
IE	A8	INTERRUPT ENABLE
IP	B8	INTERRUPT PRIORITY
P0	80	PORT 0
P1	90	PORT 1
P2	A0	PORT 2
P3	B0	PORT 3
PSW	D0	PROGRAM STATUS WORD
SBUF	99	SERIAL PORT BUFFER
SCON	98	SERIAL PORT CONTROL
SP	81	STACK POINTER
TCON	88	TIMER CONTROL
TH0	8C	TIMER 0 *<high byte>*
TH1	8D	TIMER 1 *<high byte>*
TL0	8A	TIMER 0 *<low byte>*
TL1	8B	TIMER 1 *<low byte>*
TMOD	89	TIMER MODE

PREDEFINED CODE ADDRESSES

Symbol	Hex Address	Meaning
RESET	00	Power Up (Reset)
EXTI0	03	External Interrupt 0
TIMER0	0B	Timer 0 Interrupt
EXTI1	13	External Interrupt 1
TIMER1	1B	Timer 1 Interrupt
SINT	23	Serial Port Interrupt

PREDEFINED BIT ADDRESSES

Sym.	Position	Hex	Meaning
CY	PSW.7	D7	CARRY FLAG
AC	PSW.6	D6	AUXILIARY CARRY FLAG
F0	PSW.5	D5	FLAG 0
RS1	PSW.4	D4	REGISTER BANK SELECT BIT 1
RS0	PSW.3	D3	REGISTER BANK SELECT BIT 0
OV	PSW.2	D2	OVERFLOW FLAG
P	PSW.0	D0	PARITY FLAG
TF1	TCON.7	8F	TIMER 1 OVERFLOW FLAG
TR1	TCON.6	8E	TIMER 1 RUN CONTROL BIT
TF0	TCON.5	8D	TIMER 0 OVERFLOW FLAG
TR0	TCON.4	8C	TIMER 0 RUN CONTROL BIT
IE1	TCON.3	8B	INTERRUPT 1 EDGE FLAG
IT1	TCON.2	8A	INTERRUPT 1 TYPE CONTROL BIT
IE0	TCON.1	89	INTERRUPT 0 EDGE FLAG
IT0	TCON.0	88	INTERRUPT 0 TYPE CONTROL BIT
SM0	SCON.7	9F	SERIAL MODE CONTROL BIT 0
SM1	SCON.6	9E	SERIAL MODE CONTROL BIT 1
SM2	SCON.5	9D	SERIAL MODE CONTROL BIT 2
REN	SCON.4	9C	RECEIVE ENABLE
TB8	SCON.3	9B	TRANSMIT BIT 8
RB8	SCON.2	9A	RECEIVE BIT 8
TI	SCON.1	99	TRANSMIT INTERRUPT FLAG
RI	SCON.0	98	RECEIVE INTERRUPT FLAG
EA	IE.7	AF	ENABLE ALL INTERRUPTS
ES	IE.4	AC	ENABLE SERIAL PORT INTERRUPT
ET1	IE.3	AB	ENABLE TIMER 1 INTERRUPT
EX1	IE.2	AA	ENABLE EXTERNAL INTERRUPT 1
ET0	IE.1	A9	ENABLE TIMER 0 INTERRUPT
EX0	IE.0	A8	ENABLE EXTERNAL INTERRUPT 0
PS	IP.4	BC	PRIORITY OF SERIAL PORT INTERRUPT
PT1	IP.3	BB	PRIORITY OF TIMER 1 INTERRUPT
PX1	IP.2	BA	PRIORITY OF EXTERNAL INTERRUPT 1
PT0	IP.1	B9	PRIORITY OF TIMER 0
PX0	IP.0	B8	PRIORITY OF EXTERNAL INTERRUPT 0
RD	P3.7	B7	READ DATA FOR EXTERNAL MEMORY
WR	P3.6	B6	WRITE DATA FOR EXTERNAL MEMORY
T1	P3.5	B5	TIMER/COUNTER 1 EXTERNAL FLAG
T0	P3.4	B4	TIMER/COUNTER 0 EXTERNAL FLAG
INT1	P3.3	B3	INTERRUPT 1 INPUT PIN
INT0	P3.2	B2	INTERRUPT 0 INPUT PIN
TXD	P3.1	B1	SERIAL PORT TRANSMIT PIN
RXD	P3.0	B0	SERIAL PORT RECEIVE PIN

MACRO PROCESSOR LANGUAGE (MPL) FUNCTIONS

DEFINING A FUNCTION

%*DEFINE (macro-name) (replacement-pattern)
%*DEFINE (macro-name(parameter-list)) (replacement-pattern)

MANIPULATING STRINGS

%EVAL(expression)
%LEN(string)
%SUBSTR(string,expr1,expr2)
%IN
%OUT(string)

CONTROL FUNCTIONS

%IF (expr) THEN (replacement-value) ELSE (replacement-value) FI

%REPEAT (expr) (replacement-value)

%WHILE (expr) (replacement-value)

%SET(name, value)

%MATCH(name1 name2) (list)

%MATCH(name1 delimiter name2) (string)

INTERPRETATION-CONTROLLING FUNCTIONS

%(balanced-text)

%nxxx...x
 n

%EXIT

%'comment-text'

or:

%'comment-text linefeed

%METACHAR(char)

COMPARING STRINGS LEXICALLY

%EQS
%NES
%LTS
%LES
%GTS
%GES
} (string1, string2)

ASSEMBLY TIME EXPRESSION OPERATORS

In order of decreasing precedence

( )
HIGH, LOW
*, /, MOD, SHR, SHL
+, -
EQ, LT, GT, LE, GE, NE, =, <, >, <=, >=, < >
NOT
AND
OR, XOR

ASSEMBLER DIRECTIVES

Directive	Meaning
SEGMENT	Define a relocatable segment
EQU	Assign a numeric value or special assembler symbol
SET	Set symbol value
DATA	Define a data address symbol
IDATA	Define an indirect internal data address symbol
XDATA	Define an off chip data address symbol
BIT	Assign a bit address symbol
CODE	Assign a code address symbol
DS	Reserve space in byte units
DB	Insert a list of byte values
DW	Insert a list of word values
DBIT	Advance bit location counter
PUBLIC	Declare symbols outside current module
EXTRN	List symbols defined in other modules
NAME	Identify current program module
ORG	Set location counter value
END	End of program
RSEG	Select a relocatable segment
CSEG	Select an absolute segment within code address space
DSEG	Select an absolute segment within internal data address space
XSEG	Select an absolute segment within external data address space
ISEG	Select an absolute segment within indirect internal data address space
BSEG	Select an absolute segment within the bit data address space
USING	Select predefined symbolic register banks

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	*code addr*
02	3	LJMP	*code addr*
03	1	RR	A
04	1	INC	A
05	2	INC	*data addr*
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	*bit addr,code addr*
11	2	ACALL	*code addr*
12	3	LCALL	*code addr*
13	1	RRC	A
14	1	DEC	A
15	2	DEC	*data addr*
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	*bit addr,code addr*
21	2	AJMP	*code addr*
22	1	RET	
23	1	RL	A
24	2	ADD	A,*#data*
25	2	ADD	A,*data addr*
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	*bit addr,code addr*
31	2	ACALL	*code addr*
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A,*#data*
35	2	ADDC	A,*data addr*
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R7
3F	1	ADDC	A,R7
40	2	JC	*code addr*
41	2	AJMP	*code addr*
42	2	ORL	*data addr*,A
43	3	ORL	*data addr*,*#data*
44	2	ORL	A,*#data*
45	2	ORL	A,*data addr*
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	*code addr*
51	2	ACALL	*code addr*
52	2	ANL	*data addr*,A
53	3	ANL	*data addr*,*#data*
54	2	ANL	A,*#data*
55	2	ANL	A,*data addr*

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	*code addr*
61	2	AJMP	*code addr*
62	2	XRL	*data addr*,A
63	3	XRL	*data addr*,#*data*
64	2	XRL	A,#*data*
65	2	XRL	A,*data addr*
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	*code addr*
71	2	ACALL	*code addr*
72	2	ORL	C,*bit addr*
73	1	JMP	@A+DPTR
74	2	MOV	A,#*data*
75	3	MOV	*data addr*,#*data*
76	2	MOV	@R0,#*data*
77	2	MOV	@R1,#*data*
78	2	MOV	R0,#*data*
79	2	MOV	R1,#*data*
7A	2	MOV	R2,#*data*
7B	2	MOV	R3,#*data*
7C	2	MOV	R4,#*data*
7D	2	MOV	R5,#*data*
7E	2	MOV	R6,#*data*
7F	2	MOV	R7,#*data*
80	2	SJMP	*code addr*

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
81	2	AJMP	*code addr*
82	2	ANL	C,*bit addr*
83	1	MOVC	A,@A+PC
84	1	DIV	AB
85	3	MOV	*data addr*,*data addr*
86	2	MOV	*data addr*,@R0
87	2	MOV	*data addr*,@R1
88	2	MOV	*data addr*,R0
89	2	MOV	*data addr*,R1
8A	2	MOV	*data addr*,R2
8B	2	MOV	*data addr*,R3
8C	2	MOV	*data addr*,R4
8D	2	MOV	*data addr*,R5
8E	2	MOV	*data addr*,R6
8F	2	MOV	*data addr*,R7
90	3	MOV	DPTR,#*data*
91	2	ACALL	*code addr*
92	2	MOV	*bit addr*,C
93	1	MOVC	A,@A+DPTR
94	2	SUBB	A,#*data*
95	2	SUBB	A,*data addr*
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/*bit addr*
A1	2	AJMP	*code addr*
A2	2	MOV	C,*bit addr*
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,*data addr*
A7	2	MOV	@R1,*data addr*
A8	2	MOV	R0,*data addr*
A9	2	MOV	R1,*data addr*
AA	2	MOV	R2,*data addr*
AB	2	MOV	R3,*data addr*

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
AC	2	MOV	R4,*data addr*
AD	2	MOV	R5,*data addr*
AE	2	MOV	R6,*data addr*
AF	2	MOV	R7,*data addr*
B0	2	ANL	C,/*bit addr*
B1	2	ACALL	*code addr*
B2	2	CPL	*bit addr*
B3	1	CPL	C
B4	3	CJNE	A,#*data*,*code addr*
B5	3	CJNE	A,*data addr*,*code addr*
B6	3	CJNE	@R0,#*data*,*code addr*
B7	3	CJNE	@R1,#*data*,*code addr*
B8	3	CJNE	R0,#*data*,*code addr*
B9	3	CJNE	R1,#*data*,*code addr*
BA	3	CJNE	R2,#*data*,*code addr*
BB	3	CJNE	R3,#*data*,*code addr*
BC	3	CJNE	R4,#*data*,*code addr*
BD	3	CJNE	R5,#*data*,*code addr*
BE	3	CJNE	R6,#*data*,*code addr*
BF	3	CJNE	R7,#*data*,*code addr*
C0	2	PUSH	*data addr*
C1	2	AJMP	*code addr*
C2	2	CLR	*bit addr*
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,*data addr*
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	*data addr*
D1	2	ACALL	*code addr*
D2	2	SETB	*bit addr*
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	*data addr*,*code addr*
D6	1	XCHD	A,@R0

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,*code addr*
D9	2	DJNZ	R1,*code addr*
DA	2	DJNZ	R2,*code addr*
DB	2	DJNZ	R3,*code addr*
DC	2	DJNZ	R4,*code addr*
DD	2	DJNZ	R5,*code addr*
DE	2	DJNZ	R6,*code addr*
DF	2	DJNZ	R7,*code addr*
E0	1	MOVX	A,@DPTR
E1	2	AJMP	*code addr*
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,*data addr*
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	*code addr*
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	*data addr*,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

HEX-ASCII Table

NUL	00	+	2B	V	56
SOH	01	,	2C	W	57
STX	02	-	2D	X	58
ETX	03	.	2E	Y	59
EOT	04	/	2F	Z	5A
ENQ	05	0	30	[	5B
ACK	06	1	31	\	5C
BEL	07	2	32	]	5D
BS	08	3	33	↑	5E
HT	09	4	34	←	5F
LF	0A	5	35	`	60
VT	0B	6	36	a	61
FF	0C	7	37	b	62
CR	0D	8	38	c	63
SO	0E	9	39	d	64
SI	0F	:	3A	e	65
DLE	10	;	3B	f	66
DC1/XON	11	<	3C	g	67
DC2/TAPE	12	=	3D	h	68
DC3/XOFF	13	>	3E	i	69
DC4/TAPE	14	?	3F	j	6A
NAK	15	@	40	k	6B
SYN	16	A	41	l	6C
ETB	17	B	42	m	6D
CAN	18	C	43	n	6E
EM	19	D	44	o	6F
SUB	1A	E	45	p	70
ESC	1B	F	46	q	71
FS	1C	G	47	r	72
GS	1D	H	48	s	73
RS	1E	I	49	t	74
US	1F	J	4A	u	75
SP	20	K	4B	v	76
!	21	L	4C	w	77
"	22	M	4D	x	78
#	23	N	4E	y	79
$	24	O	4F	z	7A
%	25	P	50	{	7B
&	26	Q	51	|	7C
'	27	R	52	}	7D
(	28	S	53	~	7E
)	29	T	54	DEL/RUB OUT	7F
*	2A	U	55		

intel

3065 Bowers Avenue, Santa Clara, California 95051
(408) 987-8080

Printed in U.S.A.
A1673/286/4K/DD/KH

intel *Software Development Tools*

MCS®-51 Macro Assembly Language Pocket Reference for DOS Systems

122755-001